

# **LAPORAN PRAKTIKUM**

## **Modul I**

### **Pengenalan Environment OOP**



#### **Disusun Oleh:**

Bryan Ade Bandaso

NIM :19102011

#### **Dosen**

Agus Priyanto, S.Kom., M.Kom.

**S1 Teknik Informatika**

**Institut Teknologi Telkom**

**Purwokerto**

**2020**

## **I. Tujuan Pratikum**

- a. Memahami lingkungan dasar NetBeans IDE serta cara mengcompile dan menjalankan program.
- b. Membandingkan pemrograman terstruktur dengan pemrograman berorientasi objek dengan membuat program dalam bahasa C++ dan Java.

## **II. Dasar teori**

Pemrograman Berorientasi Object atau dalam bahasa inggris lebih dikenal dengan Object Oriented Programming (OOP) adalah sebuah paradigma dalam pemrograman yang menyelesaikan masalah program dengan menyediakan objek-objek (terdiri dari beberapa *attribute* dan *method*) yang saling berkaitan dan disusun kedalam satu kelompok atau yang disebut dengan *class*. Nantinya objek-objek tersebut akan saling berinteraksi untuk menyelesaikan masalah program yang rumit.

*Apa itu objek? dan Apa itu class.* Misalkan saat anda duduk dibangku sekolah, anda pasti ditempatkan dalam satu kelas yang berisikan siswa lainnya yang sama dengan anda. Kita dapat menyimpulkan bahwa kelas itu sama dengan *class* yang kita gunakan dalam OOP dan siswa-siswi yang berada dalam kelas tersebut sama dengan objek-objek yang ada didalam *class* itu juga. Untuk memahami topik ini dengan sedikit lebih baik, terdapat beberapa istilah ataupun konsep dasar yang dapat dipahami dengan sedikit penjelasan dan contohnya. Konsep-konsep tersebut adalah :

### **Encapsulation**

Encapsulation adalah proses dalam menciptakan sebuah objek dimana terdapat beberapa bagian atau *attribute* yang terbagi berdasarkan sifat yakni *public* (umum) dan *private* (khusus).

### **Inheritance**

Saat membuat objek-objek dalam sebuah *class*, mungkin kita sering menemukan sifat yang sama antar objek dan menulisnya berulang kali sebanyak objek yang memiliki sifat tersebut. Hal itu tentu terlalu merepotkan dan memerlukan waktu yang lebih lama. Oleh karena itu terdapat suatu konsep dalam OOP yang mampu membantu kita dalam mengatasi masalah tersebut. Inheritance merupakan hubungan antara dua objek atau lebih dimana akan terdapat sebuah objek utama yang mewariskan *attribute* atau *method* yang dimilikinya kepada objek lain, baik itu keseluruhan atau sebagian.

### **Abstraction**

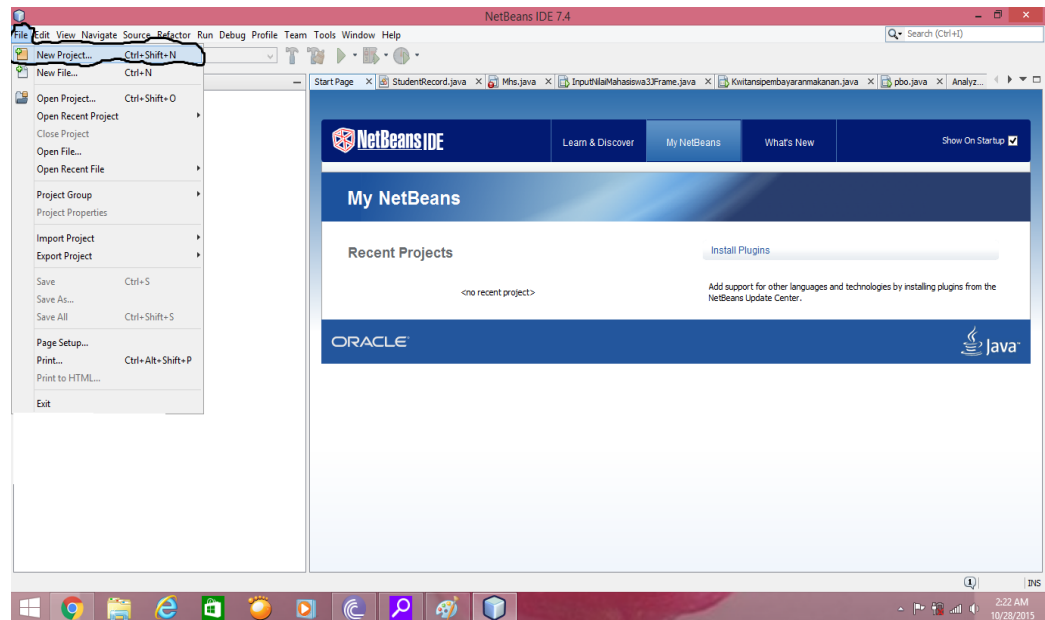
Abstraction adalah konsep dimana terdapat berbagai objek yang sejenis, namun pada saat memiliki implemementasi yang berbeda-beda. Kita dapat mengambil sebuah objek dari contoh sebelumnya seperti objek Dosen. Dosen memiliki kepentingan yang berbeda terhadap matakuliah yang tersedia dalam satu semester akademik, namun sistem akan selalu mengenali objek Dosen sebagai suatu entitas yang sama walaupun memiliki kepentingan yang berbeda-beda.

### **Composition**

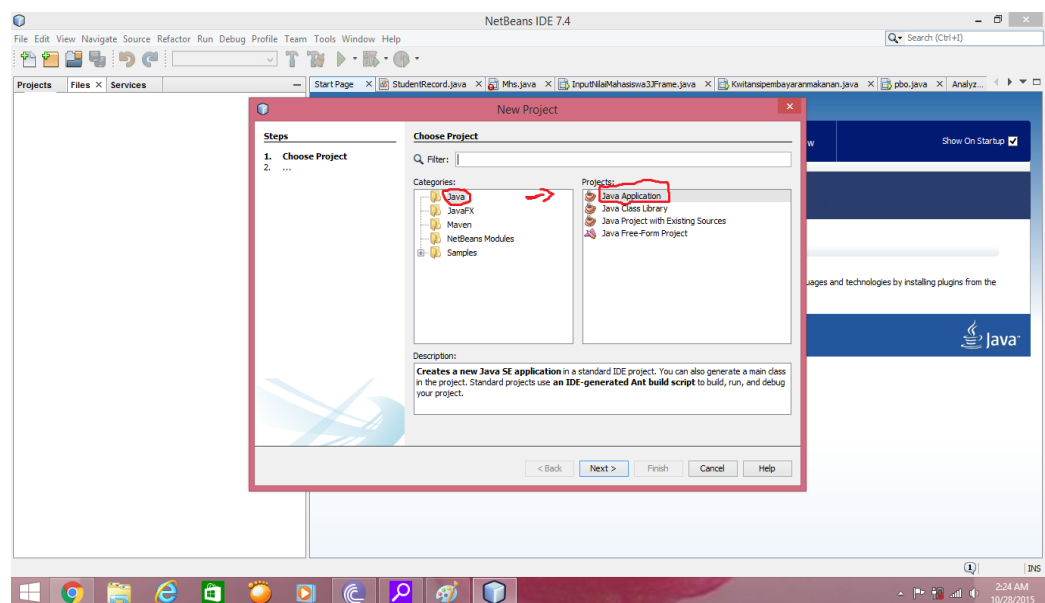
Composition adalah konsep dimana sebuah class terdiri dari beberapa bagian objek yang terpisah dan memiliki fungsi yang berbeda-beda. Contoh paling mudah untuk kita pahami adalah kembali pada contoh pertama. Tubuh manusia disusun oleh beberapa class seperti tangan, kaki, kepala dan sebagainya. Class tangan terdiri dari beberapa objek seperti Jari, Siku, dan Lengan. Kumpulan dari beberapa objek akan membangun sebuah class dan kumpulan dari beberapa class akan membangun sebuah program yang kita inginkan.

## Membuat Project

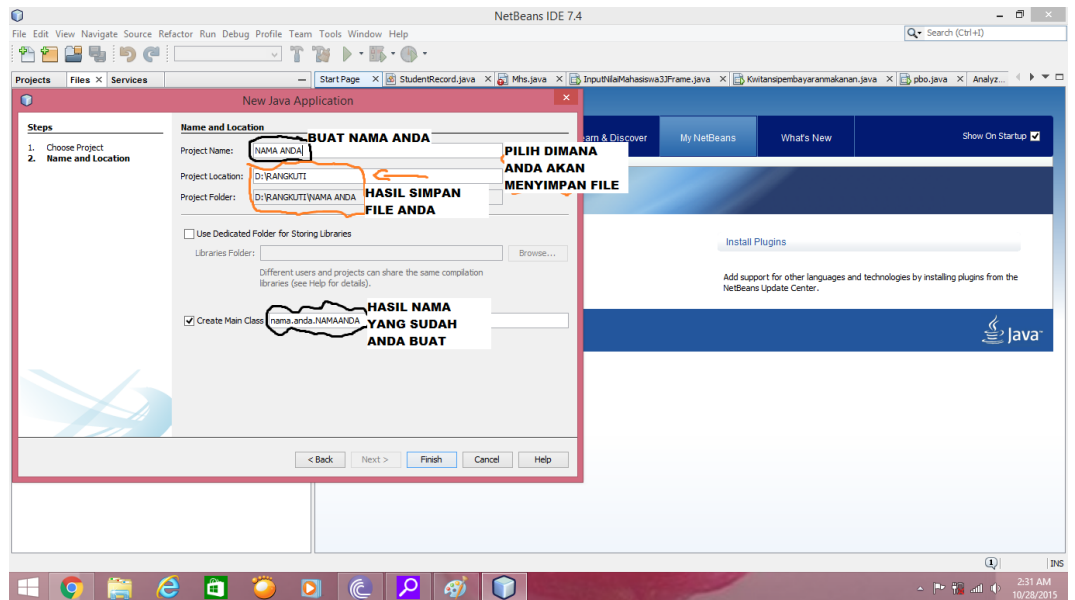
1. Buka Netbeans, pilih FILE + NEW PROJECT ( Ctrl + Shift + N )



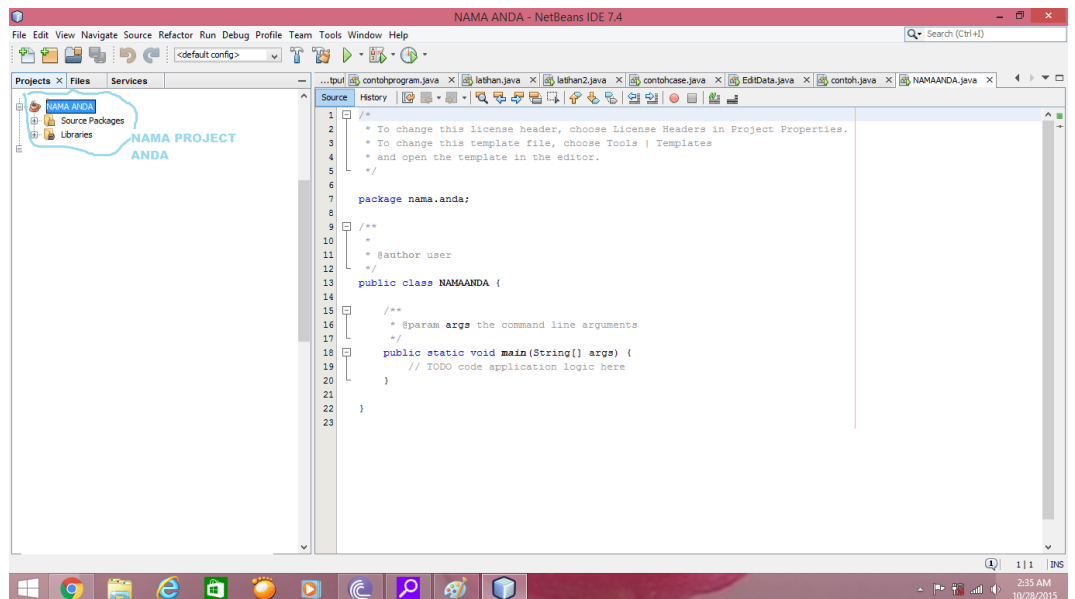
2. Java => Java Application lalu Next



4. Masukkan Nama Anda lalu klik Browser (dimana anda akan menyimpan projek anda ) setelah itu Next.

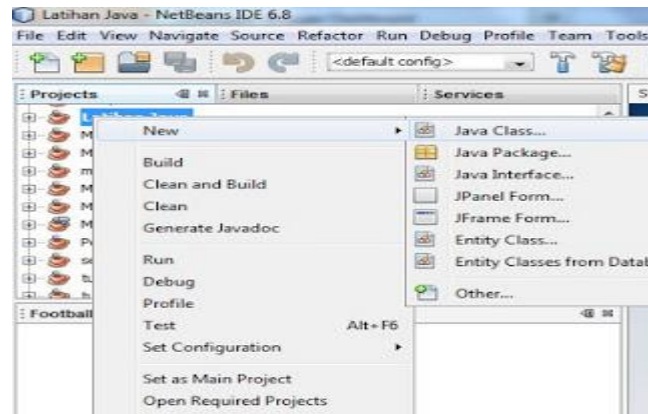


4. hasil dari Project yang anda buat tadi.

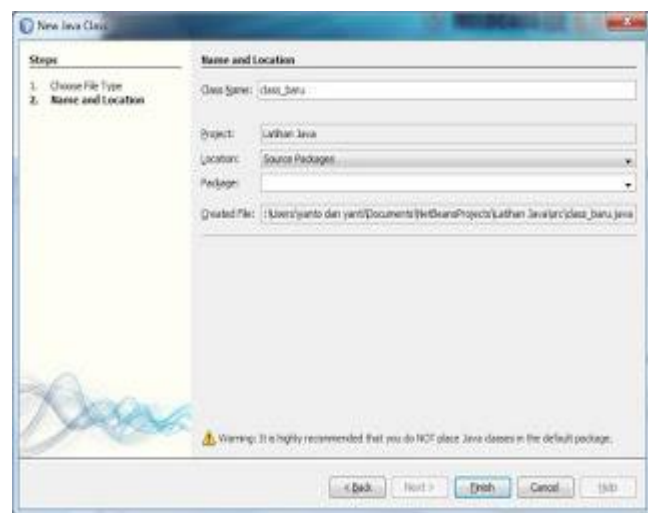


## Membuat class

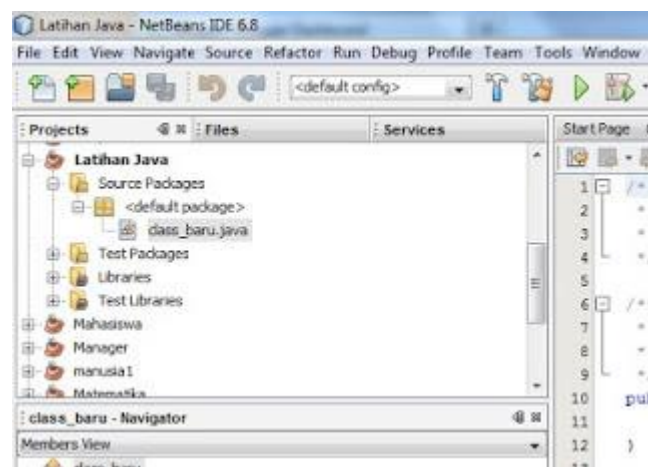
1. Buka file project hasil praktek sebelumnya, kemudian klik kanan dan pilih New >> Java Class.



2. Beri Nama class Anda kemudian klik tombol Finish. Perhatikan pada gambar dibawah ini:



3. Dibawah ini adalah contoh hasil penambahan class.



### III. Penjelasan teori

#### 1. Guide 2-1 bentuk 1

##### a. Source code

```
#include <iostream>
#include <conio.h>
// Bryan Ade Bandaso
// 19102011
using namespace std;
class Welcome
{
public:
    void display()
    {
        cout << " Selamat Datang di Praktikum PBO " << endl;
    }
};
int main()
{
    Welcome W;
    W.display();
    getch();
    return 0;
}
```

#### Guide 2-1 bentuk 2

```
#include <iostream>
#include <conio.h>
// Bryan Ade Bandaso
// 19102011
using namespace std;
class Welcome
{
public:
    void display();
```

```
};
void Welcome::display()
{
    cout << " Selamat Datang di Praktikum PBO " << endl;
}
int main()
{
    Welcome W;
    W.display();
    getch();
    return 0;
}
```

a. Hasil Output

Bentuk 1



The screenshot shows a terminal window with the title "D:\KULIAH\SEMESTER 4\Pratikum OOP\Modul\_1 C++\Guide2\_1\_bentuk1.exe". The output of the program is "Selamat Datang di Praktikum PBO".

Bentuk 2



The screenshot shows a terminal window with the title "D:\KULIAH\SEMESTER 4\Pratikum OOP\Modul\_1 C++\Guide2\_1\_bentuk2.exe". The output of the program is "Selamat Datang di Praktikum PBO".

b. Penjelasan

Pada bentuk 1 menggunakan deklarasi dan inisiasi secara langsung kelas dan fungsi display(), namun pada bentuk 2 deklarasi kelas secara langsung namun fungsi display() di inisiasi terpisah diluar kelas, sehingga kelas Welcome terbuat terlebih dahulu daripada fungsi display. Cara deklarasi dan inisiasi kelas Welcome di kedua bentuk memiliki makna atau tujuan yang sama yaitu untuk membuat suatu kelas Welcome dan memiliki fungsi display untuk menampilkan sebuah kalimat. Menurut praktikan, bentuk 1 lebih mudah dimengerti karena saat inisiasi fungsi masih di dalam blok kelas Welcome.

2. Guide 2-1

a. Source code

**Welcome.java**

```

package Guide_2_1;

/**
 * Bryan Ade Bandaso
 * 19102011
 */

public class Welcome
{
    public void display()
    {
        System.out.println("Selamat Datang di Praktikum PBO");
    }
}

Main.java

package Guide_2_1;

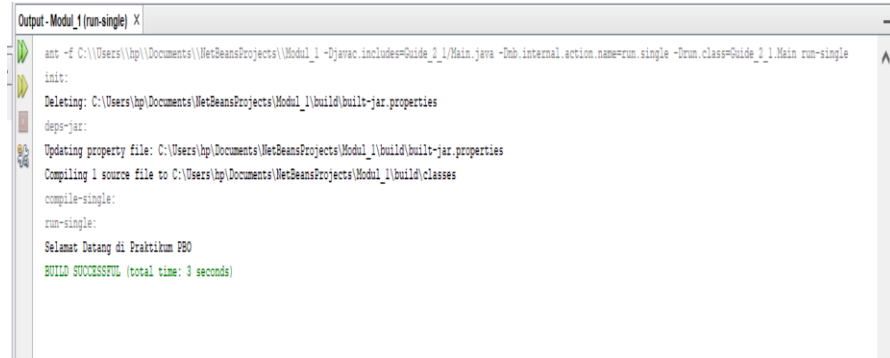
/**
 * Bryan Ade Bandaso
 * 19102011
 */

public class Main
{
    public static void main(String[] args)
    {
        Welcome W=new Welcome ();
        W.display();
    }
}

```

b. Hasil Output





```
Output-Modul_1 [run-single] X
ant -f C:\Users\hp\Documents\NetBeansProjects\Modul_1 -Djavac.includes=Guide_2_1/Main.java -Dnb.internal.action.name=run.single -Drun.class=Guide_2_1.Main run-single
init:
Deleting: C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\build-jar.properties
Compiling 1 source file to C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\classes
compile-single:
run-single:
Selamat Datang di Pratikum PBO
BUILD SUCCESSFUL (total time: 3 seconds)
```

### c. Penjelasan

Dalam source code terdapat 2 class dengan nama Welcome.java dan Main.java. didalam Welcome.java memiliki public class Welcome dan menggunakan Public void dengan nama fungsi Display(). Tipe data void berfungsi tidak mengembalikan nilai apa-apa. Dengan output nya “Selamat Datang di Pratikum PBO”.

## 3. Guide 2-2

### a. Source code

```
#include <iostream>

// Bryan Ade Bandaso
// 19102011

using namespace std;

class Mahasiswa
{
private:
    char nama[20];
    char nim[20];
public:
    void inputData();
    void display();
};

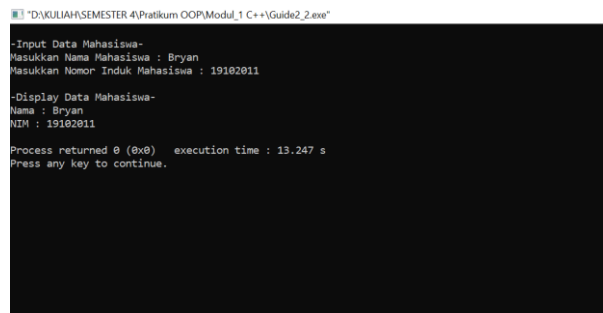
void Mahasiswa::inputData()
{
    cout << "\n-Input Data Mahasiswa-" << endl;
    cout << "Masukkan Nama Mahasiswa : ";
    cin >> nama;
    cout << "Masukkan Nomor Induk Mahasiswa : ";
```

```

        cin >> nim;
    }
    void Mahasiswa::display()
    {
        cout << "\n-Display Data Mahasiswa-"<< endl;
        cout << "Nama : " << nama << endl;
        cout << "NIM : " << nim << endl;
    }
    int main()
    {
        Mahasiswa mhs;
        mhs.inputData();
        mhs.display();
        return 0;
    }

```

#### b. Hasil Output



```

D:\KULIAH\SEMESTER 4\Praktikum OOP\Modul_1 C++\Guide2_2.exe
-Input Data Mahasiswa-
Masukkan Nama Mahasiswa : Bryan
Masukkan Nomor Induk Mahasiswa : 19102011
-Display Data Mahasiswa-
Nama : Bryan
NIM : 19102011
Process returned 0 (0x0)   execution time : 13.247 s
Press any key to continue.

```

#### c. Penjelasan

Mahasiswa dapat menggunakan bahasa pemrograman c++ mengetahui bagaimana cara menggunakan suatu variabel yang ada di dalam kelas, mengakses, dan menginputnya sesuai dengan konsep pemrograman berbasis objek.

### 4. Guide 2-3

#### a. Source code

```

Mahasiswa.java

package Guide_2_3;

/**
 * Bryan Ade Bandaso
 * 19102011
 */

```

```

import java.io.*; //Pemanggilan Library dalam Java
class Mahasiswa
{
    private String Nama[]= new String[3]; //Deklarasi array dalam Java
    private String NIM[]= new String[3];
    private int i;
    public void inputData()
    {
        BufferedReader b;
        b=new BufferedReader(new InputStreamReader
        (System.in));
        try //Penjelasan Exception lebih lanjut di berikutnya
        {
            System.out.println("-Input Data Mahasiswa-");
            for (i=0;i<3;i++) // Perulangan (looping)
            {
                System.out.print ("Masukkan Nama : ");
                Nama[i] =b.readLine (); //Pembacaan inputan melalui keyboard
                (seperti "cin" dalam C++)
                System.out.print ("Masukkan NIM : ");
                NIM[i] =b.readLine ();
            }
        }
        catch (Exception E){ }
    }
    public void display()
    {
        System.out.println("");
        System.out.println("-Display Data Mahasiswa-");
        for (i=0;i<3;i++)
        {
            System.out.println("Nama : "+Nama[i]);
            System.out.println("NIM : "+NIM[i]);
            System.out.println("");
        }
    }
}

```

```

    }
}

Main.java

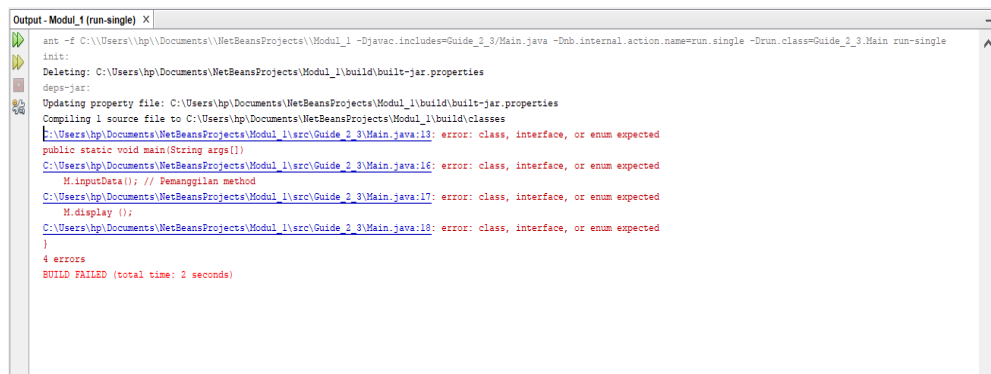
package Guide_2_3;

/**
 * Bryan Ade Bandaso
 * 19102011
 */

public static void main(String args[])
{
    Mahasiswa M=new Mahasiswa(); //Pembentukan Objek
    M.inputData(); // Pemanggilan method
    M.display ();
}

```

#### b. Hasil Output



```

Output - Modul_1 (run-single) X
ant -f C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\build-jar.properties
init:
Deleting: C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\build-jar.properties
deps-jar:
Updating property file: C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\build-jar.properties
Compiling 1 source file to C:\Users\hp\Documents\NetBeansProjects\Modul_1\build\classes
C:\Users\hp\Documents\NetBeansProjects\Modul_1\src\Guide_2_3\Main.java:13: error: class, interface, or enum expected
public static void main(String args[])
C:\Users\hp\Documents\NetBeansProjects\Modul_1\src\Guide_2_3\Main.java:16: error: class, interface, or enum expected
    M.inputData(); // Pemanggilan method
C:\Users\hp\Documents\NetBeansProjects\Modul_1\src\Guide_2_3\Main.java:17: error: class, interface, or enum expected
    M.display ();
C:\Users\hp\Documents\NetBeansProjects\Modul_1\src\Guide_2_3\Main.java:18: error: class, interface, or enum expected
}
4 errors
BUILD FAILED (total time: 2 seconds)

```

#### c. Penjelasan

Membuat sebuah program dengan 2 class dengan nama Mahasiswa.java dan Main.java. Pada Mahasiswa.java terdapat fungsi input disource codenya menggunakan BufferedReader. Dengan yang harus diinput nama, dan nim sebanyak 3 kali (fungsi looping) setelah diinput nama, dan nim akan muncul output. Untuk source code pada Main.java terdapat error dikarenakan kurang menambahkan Public class nya.

#### **IV. Kesimpulan**

1. Pratiikan harus membuat main class nya jika ingin membuat 2 class agar bisa dirunning.
2. Penggunaan public class sangat penting dalam pembuatan program.
3. Untuk Pembentukan Objek harus dengan anak classnya